```
// ----------------------------
// State Interface
// ----------------------------
interface State {
    void insertQuarter();
    void ejectQuarter();
    void turnCrank();
    void dispense();
}

// ----------------------------
// NoQuarterState
// ----------------------------
class NoQuarterState implements State {
    GumballMachine gumballMachine;

    public NoQuarterState(GumballMachine gm) {
        this.gumballMachine = gm;
    }

    public void insertQuarter() {
        System.out.println("Quarter inserted.");
        gumballMachine.setState(gumballMachine.getHasQuarterState());
    }

    public void ejectQuarter() {
        System.out.println("No quarter to eject.");
    }

    public void turnCrank() {
        System.out.println("Turned crank but no quarter.");
    }

    public void dispense() {
        System.out.println("Insert a quarter first.");
    }
}

// ----------------------------
// HasQuarterState
// ----------------------------
class HasQuarterState implements State {
    GumballMachine gumballMachine;

    public HasQuarterState(GumballMachine gm) {
        this.gumballMachine = gm;
    }
```

```
    public void insertQuarter() {
        System.out.println("You can't insert another quarter.");
    }

    public void ejectQuarter() {
        System.out.println("Quarter returned.");
        gumballMachine.setState(gumballMachine.getNoQuarterState());
    }

    public void turnCrank() {
        System.out.println("You turned the crank...");
        gumballMachine.setState(gumballMachine.getSoldState());
    }

    public void dispense() {
        System.out.println("No gumball dispensed.");
    }
}

// ----------------------------
// SoldState
// ----------------------------
class SoldState implements State {
    GumballMachine gumballMachine;

    public SoldState(GumballMachine gm) {
        this.gumballMachine = gm;
    }

    public void insertQuarter() {
        System.out.println("Please wait, dispensing gumball.");
    }

    public void ejectQuarter() {
        System.out.println("Can't eject, crank already turned.");
    }

    public void turnCrank() {
        System.out.println("Turning twice doesn’t get another gumball!");
    }

    public void dispense() {
        gumballMachine.releaseBall();
        if (gumballMachine.getCount() > 0) {
            gumballMachine.setState(gumballMachine.getNoQuarterState());
        } else {
            System.out.println("Oops, out of gumballs!");
            gumballMachine.setState(gumballMachine.getSoldOutState());
```

```
        }
    }
}

// ----------------------------
// SoldOutState
// ----------------------------
class SoldOutState implements State {
    GumballMachine gumballMachine;

    public SoldOutState(GumballMachine gm) {
        this.gumballMachine = gm;
    }

    public void insertQuarter() {
        System.out.println("Machine is sold out.");
    }

    public void ejectQuarter() {
        System.out.println("No quarter inserted.");
    }

    public void turnCrank() {
        System.out.println("Turned but no gumballs.");
    }

    public void dispense() {
        System.out.println("No gumball dispensed.");
    }
}

// ----------------------------
// GumballMachine (Context)
// ----------------------------
class GumballMachine {
    // All possible states
    private State soldOutState;
    private State noQuarterState;
    private State hasQuarterState;
    private State soldState;

    // Current state variable
    private State state;
    private int count = 0;

    public GumballMachine(int numberGumballs) {
        soldOutState = new SoldOutState(this);
        noQuarterState = new NoQuarterState(this);
```

```
    hasQuarterState = new HasQuarterState(this);
    soldState = new SoldState(this);

    this.count = numberGumballs;

    if (numberGumballs > 0) {
        state = noQuarterState;
    } else {
        state = soldOutState;
    }
}

// Actions
public void insertQuarter() {
    state.insertQuarter();
}

public void ejectQuarter() {
    state.ejectQuarter();
}

public void turnCrank() {
    state.turnCrank();
    state.dispense();
}

// Helper: release ball
void releaseBall() {
    System.out.println("A gumball comes rolling out...");
    if (count > 0) {
        count--;
    }
}

// Getters and setters
void setState(State state) {
    this.state = state;
}

public State getSoldOutState() {
    return soldOutState;
}

public State getNoQuarterState() {
    return noQuarterState;
}

public State getHasQuarterState() {
```

```
        return hasQuarterState;
    }

    public State getSoldState() {
        return soldState;
    }

    public int getCount() {
        return count;
    }
}


// ----------------------------
// Test Program (Main Method)
// ----------------------------
public class GumballTest {
    public static void main(String[] args) {
        GumballMachine gm = new GumballMachine(2);

        gm.insertQuarter();
        gm.turnCrank();

        gm.insertQuarter();
        gm.turnCrank();

        gm.insertQuarter();
        gm.turnCrank(); // Should show sold out
    }
}
```